Estándares
Desarrollo de aplicaciones del GOIB

Aplicaciones Java EE

G CONSELLERIA
O ADMINISTRACIONS
I PÚBLIQUES I
B MODERNITZACIÓ
/ DIRECCIÓ GENERAL
MODERNITZACIÓ I
ADMINISTRACIÓ DIGITAL

Palma, Febrero de 2020

Índice

C/ de Sant Pere, 7
07012 Palma
Tel. 971 17 66 00
http://dgtic.caib.es

Historial de versiones

Fecha	Versión	Descripción	Autor
05/04/16	7.2	Revisión de estándares	DGDT
17/01/20	9.0	Cambio formato documento Revisión de estándares	DGMAD
06/02/20	9.1	Corrección de errores y aclaraciones	DGMAD

C/ de Sant Pere, 7
07012 Palma
Tel. 971 17 66 00
http://dgtic.caib.es

Este documento detalla el estándar de desarrollo de aplicaciones *Java Platform, Enterprise Edition* o Java EE (recientemente renombrado como Jakarta EE) que se debe seguir para el desarrollo de aplicaciones que se instalarán en los servidores de la *Dirección General de Modernización y Administración Digital* (DGMAD).

El documento se estructura en seis grandes apartados:

1. Definición de las normas de carácter general (Capítulo 1).
2. Normativas, recomendaciones sobre ficheros de código y estructura de los proyectos (Capítulo 2)
3. Nomenclatura, requisitos, aspectos de seguridad, nombres y restricciones de las aplicaciones Java EE (Capítulo 3).
4. Versionado de código (Capítulo 4).
5. *API REST* (Capítulo 5)
6. Proceso para la generación de un nuevo proyecto Java EE basado en Proyecto Base (Capítulo 6).

Cualquier tecnología, *framework* o decisión técnica que salga de estos estándares de desarrollo deberán ser consultados y aprobados por la DGMAD. En cualquier caso, si no se realiza dicha consulta, la DGMAD se reserva el derecho de no aceptar el desarrollo o exigir el cumplimiento de estos estándares.

PROYECTO BASE

Se pone a disposición de los desarrolladores un proyecto de ejemplo con toda la tecnología, *frameworks*, estructura de proyecto, etc. que se especifican en los estándares de desarrollo de aplicaciones del GOIB.

El proyecto se pone a disposición a modo de guía para el desarrollador, no con el objetivo de convertirse en una plantilla de obligado cumplimiento en el desarrollo de aplicaciones.

¿Qué se puede encontrar en el Proyecto Base?

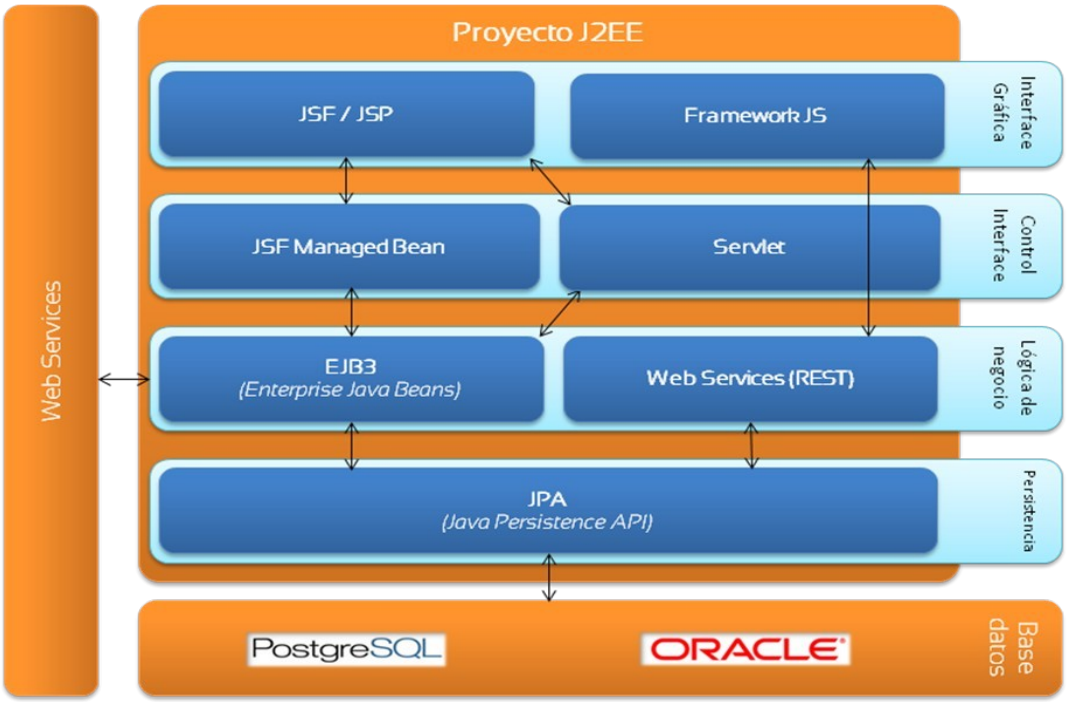
- Ejemplo de estructura de módulos y submódulos Maven.
- Ejemplo de empaquetamiento y estructura resultante esperada.
- Ejemplos de nomenclatura de clases, paquetes, métodos, tests, etc..
- Ejemplo de aplicación siguiendo la especificación Java EE 8.
- Ejemplo de publicación de servicios REST con JAX-RS (incluido en Java EE desde Java EE 6).
- Ejemplo de publicación de documentación de API REST con Swagger y Swagger-UI.

Para más información, consultar el Capítulo 6 “Cómo generar un proyecto basado en Proyecto Base”.

1. Normas de carácter general

A continuación se detallan un conjunto de normas de carácter general relacionadas con el desarrollo de aplicaciones Java EE.

- Las aplicaciones se desarrollarán siguiendo los siguientes estándares:
  - Diagramas en **UML 2.5** (de entrega obligatoria para poder realizar despliegues), especialmente:
    - Diagramas de casos de uso.
    - Diagramas de clases.
    - Diagramas de secuencia.
    - Diagramas de componentes.
    - Modelos entidad relación extendido.
  - Estándares de desarrollo de aplicaciones del GOIB (Java EE, Base de datos, implantación de aplicaciones).
  - Estándar de interfaz de usuario (libro de estilo).
- Las aplicaciones deberán desarrollar los módulos mediante aplicaciones distribuidas en tres niveles: interfaz, lógica, y datos; donde la capa de presentación se puede subdividir en capa de interfaz gráfica y capa de control de interfaz, y la capa de datos en capa de persistencia y capa de base de datos.



- Todo proyecto deberá subirse a un repositorio del GOIB diferenciando según su tipología:
  1. Aplicaciones de código libre: https://github.com/GovernIB/
  2. Aplicaciones internas: repositorio interno. Preguntar a la DGMAD.
- El producto final y las actualizaciones se entregarán según el formulario estándar de cuadernos de carga estandarizados por la DGMAD (ver Documento "Estándares. Desarrollo de aplicaciones del GOIB. Implantación de aplicaciones", Capítulo 4. "Cuaderno de carga").
- El sistema deberá cumplir las medidas y criterios de seguridad designadas en la legislación vigente, especialmente en:
  1. La Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, junto al Reglamento (UE) 2016/679 del parlamento europeo y del consejo de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento General de Protección de Datos).
  2. El Real Decreto 3/2010, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica (ENS).
  3. La Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico (LSSI).
- Así mismo, el sistema deberá cumplir las medidas de accesibilidad del Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público, junto con las WCAG (Web Content Accessibility Guidelines) en su versión 2.1.

1.1. Especificaciones tecnológicas.

A nivel general, las aplicaciones Java EE deberán implementarse utilizando la versión libre de la plataforma de desarrollo de Java **OpenJDK 11** y ejecutarse sobre un servidor de aplicaciones **JBoss EAP 7.2** sin modificar ni cambiar la configuración inicial (exceptuando lo indicado en este documento).

Respecto a la interfaz de usuario, se utilizará preferentemente **JSF 2.3** (Java Server Faces) junto a la biblioteca de componentes **Primefaces**; no obstante**,** podrá substituirse por algún **framework JS** que haya sido validado previamente por la DGMAD (como Angular o React). En ningún caso se podrá utilizar un framework propietario. Así mismo, la interfaz deberá ser compatible con la última versión estable de los navegadores **Firefox ESR** y **Chrome**.

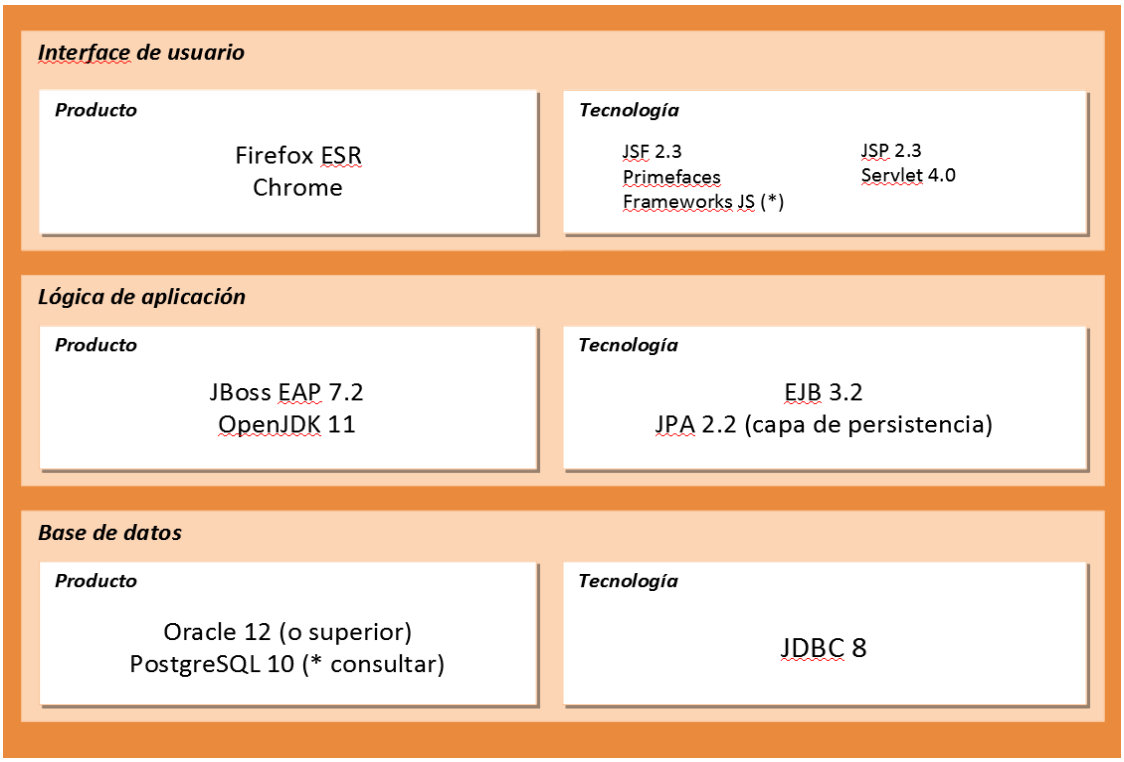
Respecto a la persistencia de datos, las aplicaciones deberán estar preparadas para funcionar sobre una base de datos **Oracle 12.2**; la necesidad de utilización de otras bases de datos como **PostgreSQL 10** deberá venir motivada y ser aprobada por parte del responsable del Área de Administración de Base de datos debido a las limitaciones de soporte y alta disponibilidad. El acceso a los datos se llevará a cabo mediante la API **JPA 2.2**.

La autenticación para acceder a los recursos protegidos de los módulos web se realiza mediante un mecanismo de autenticación centralizado: RedHat Single Sign-On 7.3 basado en **Keycloak** (ver sección 3.3 para más información).

Por último, todo proyecto Java EE deberá estar desarrollado utilizando **Maven 3.6 o superior** para la construcción del proyecto (compilación y empaquetado) y la gestión de dependencias.

En función de criterios de mantenimiento y disponibilidad de versiones, y con el objetivo de mejorar el servicio ofrecido a las consejerías, el Centro de Proceso de Datos de la DGMAD se reserva la facultad de actualizar las versiones del software aquí expuestas por otras superiores en el momento de la puesta en producción.

La siguiente imagen resume las especificaciones descritas en función del nivel de ejecución.



2. Normativa y estructura del proyecto

2.1. Normativa

Se deberá seguir:

- La codificación de los ficheros deberá ser siempre **UTF-8**.
- **Paquetes (packages)**: Los dos primeros niveles deberán ser siempre es.caib. El tercer nivel deberá indicar el nombre de la aplicación y el cuarto nivel el módulo al que pertenece. Por ejemplo: *es.caib.projectebase.back*. Todas las aplicaciones deberán de tener como mínimo cuatro niveles.

Por otro lado, se recomienda:

- **Indentación**: No usar tabulaciones. La tabulación se realizaría mediante cuatro espacios. Se pueden configurar los IDEs de programación para que emulen la tabulación.
- **Caracteres** por línea: No superar los 120 caracteres por línea.
- **Líneas por fichero**: No exceder las 1000 líneas de código por fichero. El objetivo de esta medida es distribuir el código de forma equitativa en los ficheros y dividir conceptualmente el código de forma más efectiva.
- **Documentación**: Documentar cada fichero con el formato estándar de documentación.
- **XML**: Situado al inicio del documento, con una descripción de la utilidad del fichero y en cada bloque de datos la finalidad del mismo.
- **JAVA**: Con bloques de comentarios antes de la declaración de las clases, donde se informará de la utilidad de las clases, autores y versiones. También en cada método indicando la utilidad, las variables de entrada y salida y las posibles excepciones.
- **Nomenclatura de las clases**: Identificar las clases con un nombre descriptivo relacionado con la utilidad que implemente la clase. Utilizando una o varias palabras que empiecen en mayúsculas.
- **Nomenclatura de los métodos**: En ficheros java, jsf, jsp,... seguir los estándares java: [Prefijo] + [Nombre Descriptivo]
- **Prefijo**: normalmente is, get, set , add, ... o si el método es muy sencillo, sin prefijo.
- **Nombre Descriptivo**: Una o varias palabras concatenadas sin espacio ni guiones, todas ellas comenzando en mayúsculas y describiendo la funcionalidad del método. En caso de no existir Prefijo, la primera palabra empezaría en minúscula. Ejemplos: consulta(), isAdministrador(), getEstado(), addUser(), ...

- **Nombre de campos y variables**: Formados por una o varias palabras concatenadas sin espacios ni guiones. Cada una de ellas comienza por mayúsculas excepto la primera que comienza en minúscula.
- **Nomenclatura de los test unitarios**: Identificar los test con un nombre descriptivo. Utilizar preferiblemente el mismo nombre de la clase testeada añadiendo la palabra Test al final. Ejemplo: clase a testear "MiServicio", test "MiServicioTest".
- **Constantes**: Formadas por una o varias palabras concatenadas con guiones bajos, todas ellas en mayúsculas.
- **Corchetes**: Abrir corchete en la misma línea que el comando y cerrar en una línea posterior. Ejemplo de utilización:

```
public class Exemple {
   public void getMetode(){
     try {
       for(int x=0; x < 10; x++){
          while(...) {
            if (x < 5) {
               ...
            } else {
               ...
            } // Final if-else
          } // Final while
       } // Final for
     } catch(Exception e) {
        do {
          switch(...) {
            ...
          } // Final switch
        } while(...); // Final do-while
     } // Final try-catch
   } // Final mètode
} // Final classe
```

2.2. Estructura de directorios y nombre de ficheros

A continuación se describe la estructura de directorios que deberán seguir todos los proyectos desarrollados por y para el GOIB.

Partiendo de la base que todos los proyectos GOIB deberán utilizar *Maven*, la estructura a la hora de definir los proyectos será la siguiente:

```
[Proyecto]/
    ├── pom.xml
    ├── doc/
    ├── [Módulos]/
    │     ├── [Módulo_1]/
    │     │     ├── pom.xml
    │     │     ├── src/
    │     │     │     ├── main/           (Código del módulo)
    │     │     │     │     ├── java/     (Clases java)
    │     │     │     │     ├── resources/ (Ficheros de configuración)
    │     │     │     │     └── webapp/   (Ficheros web)
    │     │     │     └── test/           (Tests unitarios)
    │     │     └── target/               (Ficheros entregables)
    │     ├── ...
    │     └── [Módulo_N]/
    ├── scripts/
    │     ├── bbdd/
    │     ├── config/
    │     ├── datasources/
    │     ├── templates/
    │     └── install/
    └── target/ (Deberá contener los entregables del proyecto)
          └── doc/
```

[Proyecto]

El módulo padre del resto de submódulos *Maven* del proyecto. En nombre de la carpeta deberá ser el nombre del proyecto. Por ejemplo: *projectebase*.

[Proyecto]/pom.xml

Fichero *Maven* encargado de la configuración, compilación y empaquetado de todo el proyecto. Se encargará de ejecutar los ficheros *pom.xml* de los submódulos del proyecto y dejar los ficheros entregables en la carpeta [Proyecto]/target.

El fichero *pom.xml* del módulo padre deberá tener ***packaging* tipo *POM*** y deberá de contener al resto de módulos. Ejemplo de fichero *POM* del *projectebase*:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>es.caib.projectebase</groupId>
<artifactId>projectebase</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
```

```
<name>Projecte Base</name>
<description>Projecte Base Java EE 8 amb Primefaces</description>
[...]
<properties>[...]</properties>
[...]
<modules>
<module>projectebase-api</module>
<module>projectebase-back</module>
<module>projectebase-commons</module>
<module>projectebase-ear</module>
<module>projectebase-ejb</module>
<module>projectebase-front</module>
<module>projectebase-persistence</module>
</modules>
[...]
</project>
```

Este fichero también contendrá todas las propiedades (*properties*) de los submódulos.

[Proyecto]/README.md

Documento sencillo donde se expliquen los requisitos, configuraciones y acciones para poder compilar, probar e instalar el proyecto.

[Proyecto]/doc

En esta carpeta se ubicarán los ficheros de documentación del proyecto. Documentos tales como diagramas, documentos de administración (instalación, configuraciones o relaciones con otros productos...), manuales de usuario, etc. Preferiblemente estarán en formato *LibreOffice*. También se deberán ubicar en esta carpeta los fuentes utilizados para la generación de los documentos, imágenes, gráficos, formatos nativos...

La documentación específica de cada submódulo, en caso de ser necesaria, se ubicará en subcarpetas con el mismo nombre que los submódulos.

[Proyecto]/doc/pdf

Aquí se dejarán los documentos anteriores convertidos en PDF. Son las versiones finales de los documentos.

[Proyecto]/[Módulos]/[Módulo N]

Colgando del módulo padre, podrá haber múltiples submódulos *Maven*. Cada uno de ellos encargado de implementar una parte de la aplicación. Como submódulos

básicos tendremos: web (*front* y/o *back*), persistencia, servicios web, lógica de negocio (*EJBs*),...

Los submódulos más comunes, con su nomenclatura propuesta son:

- *nombreaplicacion-**api**: capa de publicación de servicios REST. Atacará a la capa de lógica y/o persistencia.*
- *nombreaplicacion-**back**: capa de publicación de la aplicación web back-office.*
- *nombreaplicacion-**commons** clases y métodos con funcionalidad común a todos los módulos.*
- *nombreaplicacion-**ear**: capa para el empaquetamiento del resto de módulos (EAR).*
- *nombreaplicacion-**ejb***: capa de Enterprise Java Bean.
- *nombreaplicacion-**front***: capa de publicación de la aplicación web *front-office*.
- *nombreaplicacion-**persistence***: capa para almacenar los datos de forma persistente.
- *nombreaplicacion-**lib***: repositorio local de librerías externas. Utilizando *Maven* no deberíamos utilizar este módulo; si bien es cierto que algunas librerías no están disponibles en repositorios externos y en ocasiones no queda más remedio que añadirlas manualmente.

Nota: ver apartado 3.2. Arquitectura de aplicaciones para la estructura de módulos propuesta.

[Proyecto]/[Módulos]/[Módulo N]/pom.xml
Fichero *Maven* encargado de configurar, compilar y empaquetar el módulo.

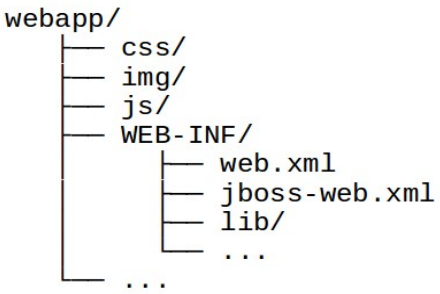
[Proyecto]/[Módulos]/[Módulo N]/src
Contendrá el código Java del módulo (*/main/java*), el código web (*/main/webapp*), el código para testear el código Java (*/test*) y todos recursos necesarios para configurar el módulo (*/main/resources*).

[Proyecto]/[Módulos]/[Módulo N]/src/main/java
Directorio donde se ubicará el código fuente del módulo. La estructura de la paquetería viene definida en el punto 2.1. *Packages*.

[Proyecto]/[Módulos]/[Módulo N]/src/main/resources
Contendrá los recursos necesarios para la aplicación. Archivos tales como: *properties*, *xml*, imágenes,...

[Proyecto]/[Módulos]/[Módulo N]/src/main/webapp
Esta carpeta contendrá todos los ficheros necesarios para especificar la *interface* web del proyecto. Estará formado principalmente por ficheros *JSF*, *JSP*, plantillas

HTML, *CSS*, *Javascript*, imágenes, ficheros de configuración, etc. La estructura básica del módulo web deberá ser:



[Proyecto]/[Módulos]/[Módulo N]/src/test/
Este directorio contendrá los tests unitarios a realizar para probar el código ubicado en "[Proyecto]/[Módulos]/[Módulo N]/src/main/java".

[Proyecto]/[Módulos]/[Módulo N]/target
En este directorio se guardarán los ficheros resultados de la compilación y empaquetamiento del módulo.

[Proyecto]/scripts/
Contendrá los *scripts* de BBDD de la aplicación. Normalmente de configuración e instalación, *datasources*, procesamiento de datos, etc.

[Proyecto]/scripts/bbdd/
Este directorio contendrá todos los *scripts* de BBDD, tanto para la creación y actualización de la estructura, como los *scripts* de inserción / eliminación / actualización de datos. El formato y estructuración de estos *scripts* seguirán el formato del Capítulo 3.5 del documento "Estándares. Desarrollo de aplicaciones del GOIB. Implantación de aplicaciones". El contenido de los scripts deberá cumplir las normas establecidas en el documento "Estándares. Desarrollo de aplicaciones del GOIB. Base de datos".

[Proyecto]/scripts/bbdd/completo
Scripts de creación de la BD desde cero a la última versión.

[Proyecto]/scripts/bbdd/[versión]
Scripts incrementales para la actualización de la BD desde la versión inmediatamente anterior a la versión [versión].

[Proyecto]/scripts/config/
Scripts de configuración del sistema y del producto.

[Proyecto]/scripts/datasources/

En esta carpeta se ubicarán los ficheros utilizados por *JBoss* para acceder a la BBDD. Dichos ficheros contienen la información necesaria para conectarse a una BBDD: *driver* de conexión, tipo de BD, *host*, puerto, nombre de la BBDD, usuario, contraseña, etc.

[Proyecto]/scripts/templates/

En este directorio se guardarán las plantillas para generar ficheros. Por ejemplo, para generar los ficheros de versión que se explican en el punto 4.3.

[Proyecto]/target/

Directorio donde se ubicará el producto final encapsulado y listo para el despliegue. En esta carpeta deberá haber el/los ficheros *EAR* o *JAR* a desplegar.

[Proyecto]/target/doc

En esta carpeta se deberán ubicar los documentos asociados al proyecto empaquetado. Por ejemplo:

- *javadocs*: Documentos generados a partir de las clases Java. Es por ello que será imprescindible documentar correctamente cada clase siguiendo dicha especificación.
- Otros documentos que se considere necesario.

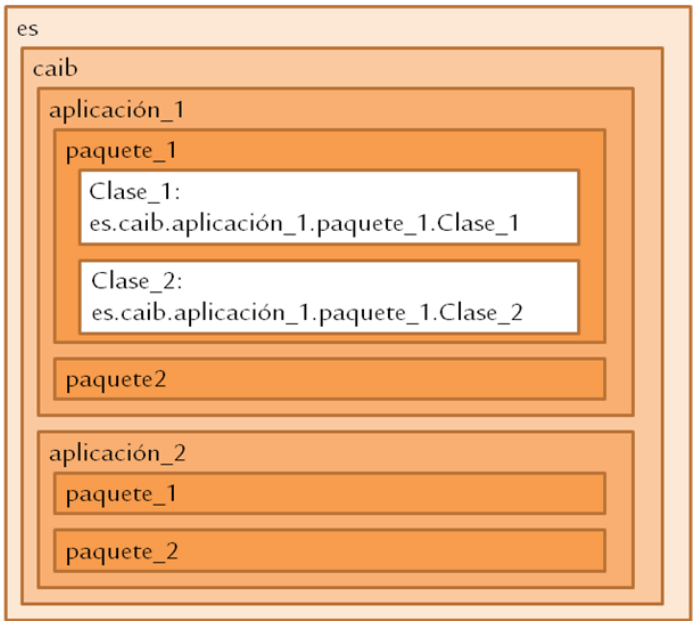
3. Normativa de las aplicaciones Java EE

3.1. Nomenclatura

3.1.1. Jerarquía de paquetes

Las clases de objetos se estructurarán en aplicaciones y paquetes. Todas las aplicaciones y paquetes dependerán de forma jerárquica del dominio de paquetes *es.caib*.

Así, las clases se denominarán *es.caib.nombreaplicación.paquete.Clase*, tal y como puede observarse en la siguiente ilustración:



Los caracteres válidos serán aquellos definidos por el estándar Java: letras mayúsculas y minúsculas del alfabeto inglés y números en posición no inicial.

Los nombres de aplicación estarán siempre en minúsculas y deberán ser solicitados y autorizados por el Centro de Proceso de Datos de la DGMAD (ver documento “Estándares. Desarrollo de aplicaciones del GOIB. Implantación de aplicaciones”, Capítulo 2. “Solicitud de código de aplicación”).

Los nombres de paquete estarán siempre en minúsculas y podrán ser nombrados, dentro del paquete de aplicación, a criterio de analistas y diseñadores.

3.1.2. Nomenclatura de clases

Las clases se nombrarán con la primera letra mayúscula y el resto en minúsculas.

Las clases formadas por varias palabras utilizarán mayúsculas para la inicial de cada una de ellas.

Ejemplos:

- *es.caib.aplicacion.paquete.Clase*
- es.caib.aplicacion.paquete.ClaseDeVariosVocablos

3.1.3. Nomenclatura de métodos

Los métodos se nombrarán con todas las letras minúsculas, incluida la inicial.

Las clases formadas por varias palabras utilizarán mayúsculas para la inicial de las segundas palabras.

Ejemplos:

- es.caib.aplicacion.paquete.Clase.metodo
- es.caib.aplicacion.paquete.Clase.metodoDeVariosVocablos

3.1.4. Servicios de directorio de servidor de aplicaciones

El acceso al servicio de directorio (*NamingFactory*) se realizará siempre con los parámetros por defecto, asumiendo que las propiedades *JNDI* están correctamente configuradas.

Los servicios de directorio del servidor transaccional identificarán cada *Enterprise Java Bean* mediante su nombre jerárquico completo, debiendo acceder las clases Java a él mediante dicho nombre.

El acceso a otro tipo de servicios, tales como conexiones a base de datos o *pools* de conexiones se realizará a través de nombres jerárquicos dependientes de la jerarquía de la aplicación.

Ejemplo:

- Base de datos: *es.caib.aplicación.db*
- Mail: *es.caib.aplicacion.mail*

3.1.5. Acceso a bases de datos

El acceso a bases de datos se realizará a través de la capa de persistencia, definida en el fichero ***persistence.xml***. Dicho acceso se realizará mediante la unidad de persistencia definida en dicho fichero.

Los ficheros de *datasource* deben tener el formato ***nombreAplicacion-ds.xml***, por ejemplo: *projectebaseexemple-ds.xml*. Dentro del *datasource*, se debe añadir la siguiente línea resaltada para establecer un esquema por defecto:

```
<datasource jndi-name="java:jboss/datasources/codigoAppDS" pool-name="codigoAppDS"
enabled="true" use-java-context="true">
        <connection-url>jdbc:oracle:thin://host:1521/nombrebd</connection-url>
        <driver>oracle</driver>
        <security>
          <user-name>WWW_CODIGOAPLICACION</user-name>
          <password>contraseña</password>
        </security>

        <new-connection-sql>
        BEGIN
        EXECUTE IMMEDIATE 'ALTER SESSION SET CURRENT_SCHEMA = NOMBREBD';
        END;
        </new-connection-sql>
</datasource>
```

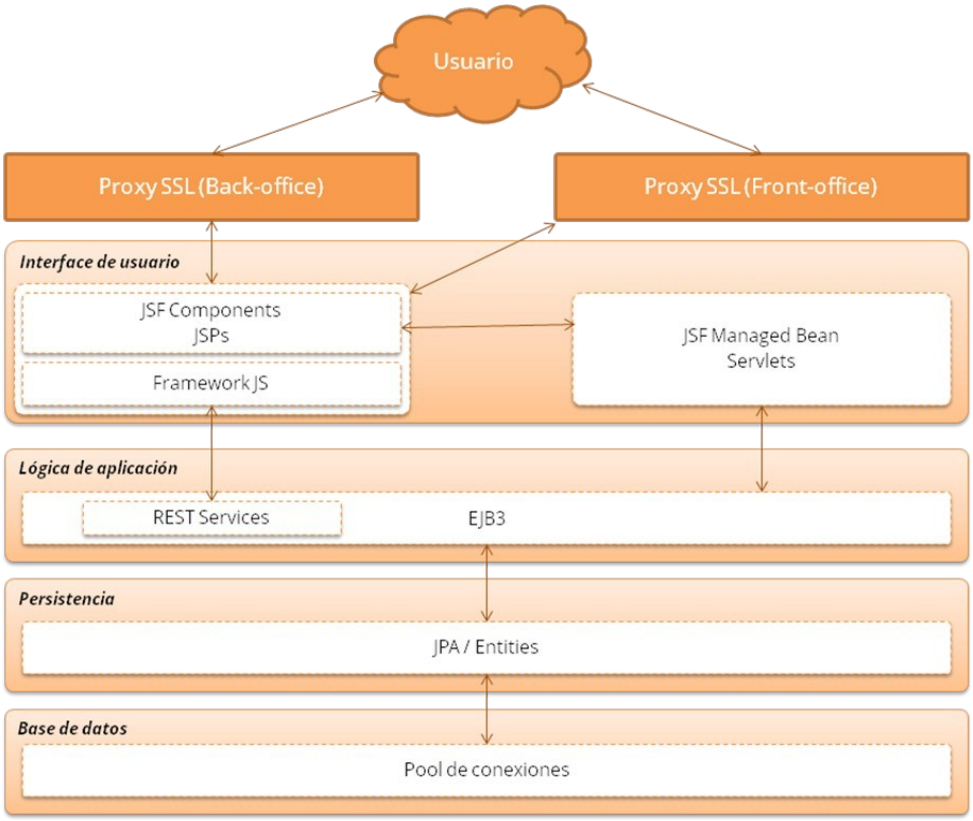
El usuario del *pool* de conexiones deberá seguir la nomenclatura WWW_CODIGOAPLICACION.

En *Oracle*, el acceso a la base de datos debe hacerse utilizando cliente thin (no OCI).

3.2. Arquitectura de aplicaciones

La arquitectura de la aplicación deberá ser la siguiente, si bien se pueden admitir ligeras variantes:

Normalmente, la petición del usuario será recogida por un *servlet* o controlador *JSF*, el cual localizará el *Enterprise Java Bean* adecuado a través de su nombre *JNDI* o preferiblemente mediante la inyección del mismo. Una vez disponible se solicitará al *EJB* la ejecución de las acciones pertinentes.



Bajo ninguna circunstancia ni el *servlet*, ni el controlador ni las páginas *JSP/JSF/HTML* deberán acceder de forma directa a la base de datos. Tampoco estará permitido el acceso directo desde las páginas hacia los *EJB*. La única excepción será el acceso desde páginas gestionados con *frameworks* JS hacia los servicios *REST*.

Toda operación contra bases de datos deberá ser canalizada a través de los *EJBs* y estos, a su vez, a través de las entidades de persistencia gestionadas por *JPA*.

Estructura propuesta de los módulos *Maven*

Las aplicaciones deberán dividirse en módulos *Maven*. Como mínimo los proyectos deberán estar formados por un proyecto *POM* padre que contenga (como mínimo) un módulo *WAR*, un módulo *JAR* y un módulo *EAR*. A continuación se puede ver un ejemplo de fichero *POM* de cada uno de los tipos de módulo *Maven* propuestos:

a) **Un módulo padre con *packaging* tipo *POM*** que deberá contener al resto de módulos.

b) **Uno o varios módulos Web (*Frontal / Backoffice*) con *packaging WAR*** que contengan todo el código de la capa de presentación: Control de interfaz e

interfaz. En el caso de *JSF Managed Beans* y páginas *XHTML*. Los nombres por defecto serán ***códigoAplicación-front-versión.war*** y ***códigoAplicación-back-versión.war***.

c) **Un módulo con la lógica de aplicación (*EJB3*) y la capa de persistencia (*JPA*) con *packaging JAR*.** El nombre por defecto deberá ser ***códigoAplicación-version.jar***.

d) Un módulo de empaquetamiento del resto de módulos con *packaging EAR*. El nombre por defecto deberá ser ***códigoAplicación.ear***.

3.3. Seguridad de aplicaciones

Todos los aspectos relativos a identificación y autorización de los usuarios a *servlets*, *REST*, controladores, *JSPs* o *EJBs* estarán gestionados de forma externa a las aplicaciones, desde el entorno de administración de la plataforma *Java EE*, por lo que no se debe codificar dentro de *servlets*, *REST*, controladores, *JSPs* o *EJBs* ninguna regla o criterio de autenticación. Sí pueden estar codificados dentro de la aplicación aspectos relativos a cómo se presenta el *interface* de usuario.

En caso de que la aplicación requiera restringir el acceso a los recursos mediante un usuario y contraseña deberán configurarse los elementos **security-constraint**, **login-config**, y **security-role** dentro del fichero **web.xml** (ver sección 3.3.4).

3.3.1. Control de accesos

A partir de la versión *EAP* 7.2 de *JBoss*, la autenticación para acceder a los recursos protegidos de los módulos web se realiza mediante un mecanismo de autenticación centralizado: *RedHat Single Sign-On (Keycloak)*. El propio servidor de la aplicación (*JBoss*), al recibir una petición de acceso a un recurso que requiera autorización, se encargará de redirigir al usuario a la página de autenticación.

Dicho módulo de autenticación permitirá el control de accesos a las aplicaciones a nivel de *war* o módulo web. Por lo tanto, toda aplicación deberá estar separada en distintos módulos que requieran distintos mecanismos de autenticación. Éstos se configuran en *Keycloak* como *clients* y se indican en el fichero *standalone.xml* del JBoss como *resources*:

- **goib-default**: permite al usuario autenticarse con certificado digital (cualquiera de los admitidos por *@firma*) o, si no dispone de certificado, mediante su usuario y contraseña del GOIB.

- **goib-ws**: se trata del mecanismo de autenticación que debe aplicarse para proteger módulos que contengan *servicios REST*. Básicamente, permite autenticación *BASIC*, sin redirigir la petición a la web centralizada de autenticación.
- **goib-cert**: solo habilita la autenticación con certificado, con lo cual, solo se podrá acceder a los contenidos protegidos del módulo, mediante el uso de un certificado válido.

3.3.2. Instalación y configuración del adaptador de RHSSO

Para utilizar el nuevo módulo de autenticación es necesario instalar el adaptador de *RedHat Single Sign-On* sobre *JBoss*, para ello hay que descargar el adaptador de *RH-SSO 7.3* y aplicarlo a la instancia de *JBoss*:

```
$ cd $JBOSS_HOME\bin
$ unzip rh-sso-7.3.0.GA-eap7-adapter.zip
$ ./jboss-cli.sh –file=adapter-install-offline.cli
```

Este adaptador instalará un módulo o subsistema en el fichero *standalone.xml*. Toda la información resaltada será proporcionada por la DGMAD.

Hay que diferenciar la *securización* de los módulos *front* y *back* de los módulos *API* ya que difieren en la inyección de credenciales del usuario empleado.

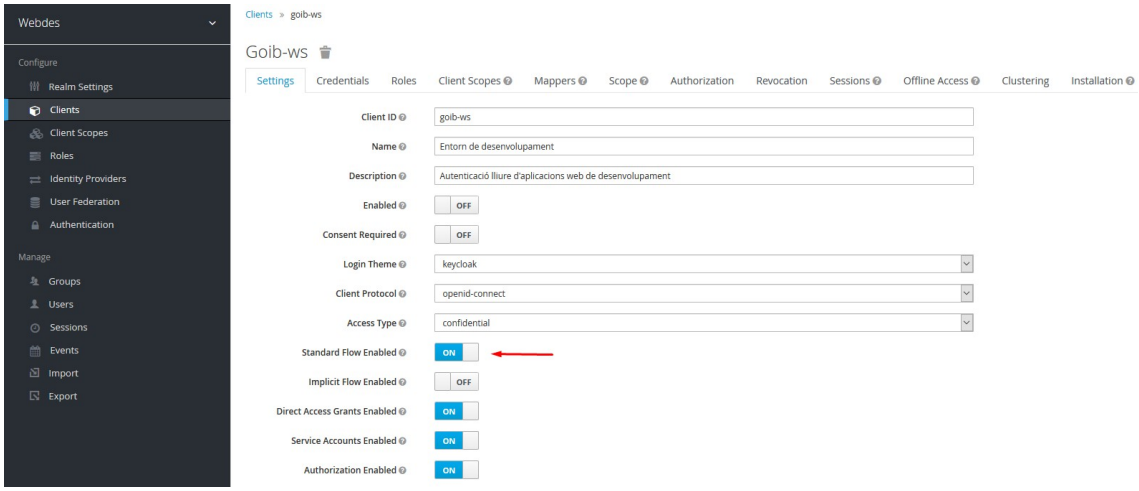
En el caso de un módulo *front* o *back* se configuraría de la siguiente manera:

```
<subsystem xmlns="urn:jboss:domain:keycloak:1.1">
        <realm name="NOM_REALM">
            <auth-server-url>URL_KEYCLOACK</auth-server-url>
            <ssl-required>MODE</ssl-required>
        </realm>
        <secure-deployment name="NombreDelWar1.war">
            <realm>NOM_REALM</realm>
            <resource>goib-XXXXX</resource>
            <use-resource-role-mappings>false</use-resource-role-mappings>
            <public-client>true</public-client>
            <verify-token-audience>true</verify-token-audience>
        </secure-deployment>
        <secure-deployment name="NombreDelWar2.war">
        ....
        </secure-deployment>
</subsystem>
```

En el caso de una *API REST:*

```
<secure-deployment name="NombreDelWarDeWS.war ">
    <realm>NOM_REALM</realm>
    <auth-server-url>URL_KEYCLOAK</auth-server-url>
    <resource>goib-ws</resource>
    <use-resource-role-mappings>true</use-resource-role-mappings>
    <bearer-only>true</bearer-only>
    <enable-basic-auth>true</enable-basic-auth>
    <ssl-required>MODE</ssl-required>
    <credential name="secret">CREDENTIAL_SECRET</credential>
</secure-deployment>
```

En este último caso, será necesario activar la opción *Standard Flow Enabled* del *cliente* en la configuración del Keycloak.



3.3.3. Elemento <security-role> del fichero web.xml

En el fichero *web.xml* (y *ejb-jar.xml*) se deberán definir uno o varios roles para la aplicación, con sus respectivas descripciones.

Ejemplo:

```
<security-role>
<description> ... descripción ...</description>
<role-name>APL_ XXXXXX</role-name>
</security-role>
```

Para poder integrar la seguridad definida a nivel de aplicación con el sistema de seguridad del GOIB será necesario que los nombres de roles definidos en el fichero *web.xml* estén estandarizados según las normas de la DGMAD.

Para el caso de una aplicación con prefijo *APL_* el nombre especificado con el *tag* <*role-name*> debe ser *APL_XXXXXX*, donde *XXXXXX* debe ser un nombre lo más simple y representativo posible.

Ejemplos de nombres de roles:

- *APL_CONSULTA*
- *APL_INTRODUCCIO*
- APL_ADMINISTRACIO

3.3.4. Elemento <security-constraint>

Se deberá utilizar en caso de tener que definir privilegios de acceso para una colección de recursos. Deberán especificarse los roles que tendrán acceso a los recursos protegidos.

A continuación se muestra un ejemplo de configuración para restringir el acceso al entorno de backoffice del Proyecto Base:

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Todos los recursos</web-resource-name>
      <description>Todos los recursos</description>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>PB_ADMIN</role-name>
   </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>Govern de les Illes Balears</realm-name>
</login-config>
<security-role>
   <role-name>PB_ADMIN</role-name>
</security-role>
```

3.3.5. Protección de EJBs

Es necesario proteger los *EJBs* de manera que ningún usuario anónimo pueda ejecutarlos, salvo que los *EJBs* deban ser públicos. Para protegerlos se deberá utilizar la anotación ***@RolesAllowed*** pertinente en los métodos o clases que se deseen proteger.

Ejemplo:

```
@RolesAllowed({"PB_ADMIN"})
public class FooService implements FooServiceInterface {
...
}
```

En el caso de que los EJBs deban ser públicos, no se deberá especificar dicha anotación *@RolesAllowed.*

3.3.6. Declaración de dominios de seguridad en JBoss (<security-domain>)

No se deberá especificar ningún *security-domain*, ni declarativamente ni en los descriptores *jboss-jar.xml, jboss-web.xml* o *jboss.xml.* Si dichos ficheros solo se usan para indicar el *security-domain,* éstos se pueden eliminar directamente.

Asimismo, tampoco se deberá especificar cualquier referencia al *security-domain* a través de anotaciones en el código fuente (*@SecurityDomain*).

3.4. Nomenclatura de módulos y contextos web

Como normal general las aplicaciones estarán compuestas, como mínimo por un módulo de capa web (*war*) y un módulo de capa lógica (*jar*), en tal caso los nombres de dichos módulos serán:

- <códigoAplicación>-versión.war
- <códigoAplicación>-versión.jar

En tal caso, el contexto de acceso a la aplicación tendrá que ser */<código_aplicación>*, con lo cual todos los recursos de la aplicación tendrán que estar disponibles bajo dicho contexto.

Dependiendo de los requisitos de acceso a la aplicación (por ejemplo si se requiere diferenciar entre acceso público y *backoffice*) y la configuración mencionada no es suficiente, se podrán añadir los siguientes módulos *war* siguiendo las siguientes pautas:

1. Módulo de *backoffice*: Anexando el sufijo *back* detrás del código de aplicación:
- Nomenclatura del fichero: *códigoAplicación-**back**-versión.war*
- Contexto del módulo: */códigoAplicaciónback*
- *module-name* en *web.xml*: *códigoAplicaciónback*

2. Módulo de *frontoffice*:
- Nomenclatura del fichero: *códigoAplicación-**front**-versión.war*
- Contexto del módulo: */códigoAplicaciónfront*
- *module-name* en *web.xml*: *códigoAplicaciónfront*

3. Módulo de *api/[rest|ws]*:
- Nomenclatura del fichero: *códigoAplicación-**api**-versión.war*
- Contexto del módulo: */códigoAplicación/api*
- *module-name* en *web.xml*: *códigoAplicación**api***

En caso de que fuera necesario diferenciar entre *webservices* de acceso público y de *backoffice* se podrían diferenciar anexando *back* y *front* respectivamente, es decir:

- códigoAplicación**api**back
- códigoAplicación**api**front

3.5. Restricciones adicionales

Para todas aquellas funcionalidades no especificadas en este documento, se recomienda la utilización de estándares *Java EE* en su versión 8 o, en su defecto, en su versión 7, evitando en la medida de lo posible soluciones particulares que solo funcionen sobre *JBoss*.

- Las aplicaciones deberán utilizar el juego de caracteres ***UTF-8***: *<?xml version=... encoding="UTF-8" ?>*
- Los mensajes de depuración generados por la aplicación deberán aparecer en la categoría *DEBUG*, nunca *INFO* o *WARN* (usando siempre log4j).
- Los *beans* serán preferentemente *stateless session beans*.
- Los *stateful session beans* deberán implementar adecuadamente los métodos *activate* y *passivate* al efecto de minimizar el consumo de memoria y recursos.

3.5.1. Implantación de buenas prácticas en el código.

El equipo de desarrollo de la aplicación deberá aplicar reglas de buenas prácticas generalmente reconocidas con el objetivo de reducir la aparición de errores en el código cuando el sistema se lleve a producción.

Algunas de estas buenas prácticas pasan por:
- Priorizar la legibilidad del código.
- Minimizar el acoplamiento (dependencia) entre módulos.
- Generar y ejecutar tests unitarios de todas aquellas clases que incluyan lógica de negocio.
- Evitar el uso de código redundante.
- Cerrar correctamente los ficheros y conexiones cuando ya no se necesiten.

- Eliminar variables, constantes, e importaciones de clases que no se utilicen.
- Establecer un correcto sistema de control de errores.
- Documentar y comentar adecuadamente el código.

Actualmente, la mayoría de entornos de desarrollo integrado, en inglés *Integrated Development Environment* (IDE), incluyen herramientas nativas o extensiones que facilitan la labor del desarrollador en este sentido.

3.5.2. Propiedades de configuración

Las aplicaciones podrán hacer uso de propiedades para parametrizar diferentes aspectos de la aplicación mediante el uso de dos ficheros *.properties:*

- ***códigoAplicación.properties***: de propiedades internas de cada aplicación. La DGMAD en ningún caso se hará cargo de añadir propiedades o modificar los valores de este fichero. Se deberá mandar a despliegue el fichero entero que ya incorpore los cambios respecto al anterior.
- ***códigoAplicación.system.properties***: de propiedades configurables por parte de la DGMAD. En este caso, la DGMAD sí que se encargará de configurar dicho fichero, que se usará exclusivamente para las siguientes propiedades (de uso opcional):
  - **es.caib.códigoAplicación.fitxers**: previa solicitud, apuntará a un directorio en el cual la aplicación podrá guardar ficheros. En dicho directorio, la aplicación podrá crear la estructura que mejor se adapte a sus necesidades. Si fuera necesario crear una estructura con diferentes subdirectorios, éstos deberán crearse bajo esa ruta, y se deberán definir las propiedades necesarias relativas al directorio base apuntado por es.caib.códigoAplicación.fitxers en el fichero códigoAplicación.properties.
  - **es.caib.códigoAplicación.int.nombreIntegración.usuari** y
  - **es.caib.códigoAplicación.int.nombreIntegración.secret**: para los casos en que la aplicación tenga que realizar llamadas a webservices, la DGMAD configurará el usuario y el password, respectivamente en dichas propiedades. En el caso de usuarios de aplicación, nombreIntegración tiene el formato aplicacionOrigen_aplicacionDestino.
  - **es.caib.códigoAplicación.int.nombreIntegración.endpoint**: Url que apunta al servicio de aplicación que se desea llamar.

- **es.caib.códigoAplicación.int.nombreIntegración.path**: Path hacia la integración deseada. Ej: path hacia ruta de ficheros estáticos: es.caib.codigo.int.staticfiles.path=C:/Desarrollo.

Las aplicaciones podrán cargar los ficheros de *properties* a través de las propiedades de *System*:

- es.caib.códigoAplicación.properties
- es.caib.códigoAplicación.system.properties

A continuación se muestra un ejemplo de código para cargar las propiedades de los diferentes ficheros *properties:*

```
try (InputStream input = new
FileInputStream(System.getProperty("es.caib.codigoAplicación.properties")) {
        Properties prop = new Properties();
        // load a properties file prop.load(input);

        // get the property value and print it out
        String x = prop.getProperty("es.caib.codigoAplicacion.xxxx");
        String y = prop.getProperty("es.caib.codigoAplicacion.yyyy");
        String z = prop.getProperty("es.caib.codigoAplicacion.zzzz");

} catch (IOException ex) {
        ex.printStackTrace();
}

try (InputStream input = new
FileInputStream(System.getProperty("es.caib.codigoAplicación.system.properties")) {
        Properties propDGMAD = new Properties();
        // load a properties file propDGMAD.load(input);

        // get the property value and print it out
        String directoriBaseFitxers =
propDGMAD.getProperty("es.caib.codigoAplicacion.directori");

} catch (IOException ex) {
        ex.printStackTrace();
}
```

Las propiedades tendrán como prefijo *es.caib.codigoAplicacion.* Por ejemplo, el contenido del fichero *codigoAplicacion.properties* podría ser:

```
es.caib.codigoAplicacion.propiedad1=valor1
es.caib.codigoAplicacion.propiedad2=valor2
es.caib.codigoAplicacion.propiedad3=valor3
```

4. Versionado de código

4.1. Referencia a la versión en los ficheros generados

Todos los ficheros generados (*WAR*, *JAR*, ...) deben incorporar la versión del producto. Para ello hay que definir el *finalName* en los ficheros *pom.xml* de todos los módulos de la siguiente forma:

```
<build>
<finalName>projectebase-${project.version}</finalName>
...
</build>
```

4.2. Incluir información de versionado en el fichero MANIFEST.MF

Será necesario incluir información de versionado, fecha y revisión en el fichero *MANIFEST.MF* del *EAR*.

4.3. Mostrar la versión del producto durante la ejecución del producto

Para mostrar la información de versión de la aplicación, en *logs*, páginas, etc. Será obligatorio generar clases necesarias para ello:

4.3.1. Crear plantilla de la clase del versionado

Crear el fichero ***Version.java.template*** en [Proyecto]/scripts/templates con el siguiente contenido:

```
package es.caib.projectebase;
// Código autogenerado por la Version.java.template.
public final class Version {
	public static final String VERSION="@project.version@";
	public static final String BUILDTIME="@project.buildtime@";
	public static final String VCS_REVISION="@vcs.revision@";
	public static final String JDK_VERSION="@jdk.version@";
}
```

4.3.2. Hacer referencia a la versión en las páginas o clases Java

Para hacer referencia a la versión y fecha de generación en las páginas se puede hacer invocando algún *Servlet*, Controlador *JSF*, utilizando *<%=Version.VERSION%>*

en *JSP*, etc, que utilice la clase *Version.java* generada anteriormente y que introduzca en el pie de la página la versión y la fecha de generación del producto.

En cuanto a las referencias en clases java se podrá hacer utilizando *Version.VERSION*, siempre y cuando se haya definido la clase *Version.java* a partir de la plantilla *Version.java.template*.

4.4. Mostrar la versión en el log *de aplicación*

Al arrancar la aplicación de deberá imprimir un log (utilizando el *logger* a nivel *INFO*) con el nombre del producto y la versión que se está ejecutando así como la fecha de generación. Por ejemplo:

```
LOGGER.info("Cargando la aplicación PROYECTOBASE versión "+Version.VERSION+ "
generada en fecha: "+Version.BUILDTIME);
```

5. API REST

Es obligatorio la publicación de una *API REST* donde se puedan consultar todos los datos susceptibles de ser reutilizados por otras aplicaciones o para ser publicados en formatos abiertos para su posible reutilización por usuarios externos al GOIB.

Se **restringe entonces el uso de servicios *SOAP*** en favor de servicios *REST*. Si se desean publicar servicios *SOAP* se deberá consultar con la DGMAD y justificar su utilización.

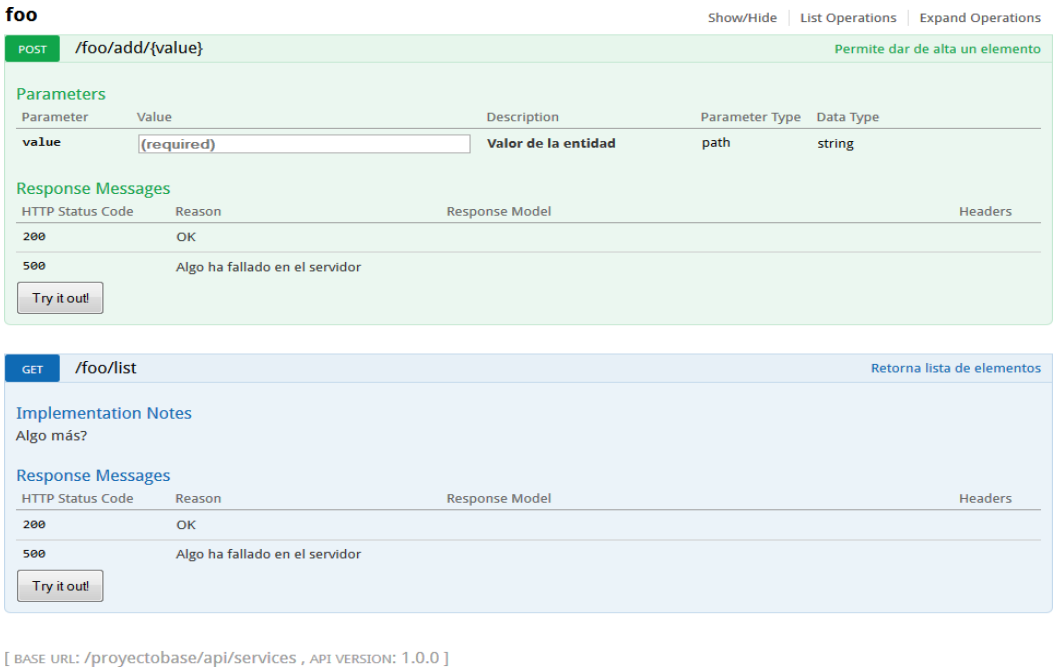
Para la publicación de las *API REST* se deberán de seguir las siguientes pautas:

1) Se deberá crear un submódulo *Maven* llamado *nombreaplicacion-**api*** bajo el proyecto padre donde se ubicarán los recursos del *API*.
2) El API se deberá publicar siempre en el contexto */nombreaplicacion/api/services/**
3) Los datos deberán devolverse siempre en **formato *JSON* o *CSV*** (dependiendo de la finalidad de los datos).
4) Se deberá generar una documentación del *API* de forma automática mediante una herramienta tipo *Swagger* (ver ProjecteBase para ver un ejemplo). Será opcional la utilización de una herramienta visual tipo *Swagger-UI*. El contexto de publicación deberá ser siempre */nombreaplicacion/api/swagger.json* o */nombreaplicacion/api/index.html*. Y cualquiera de los casos deberá ser la página de bienvenida de la aplicación en el contexto */nombreaplicacion/api*.

Ejemplo de *welcome file* en *web.xml*

```
<web-app>
...
<welcome-file-list>
<welcome-file>apidoc.jsp</welcome-file>
</welcome-file-list>
...
</web-app>
```

Ejemplo de documentación *API* publicada con *Swagger-UI*:



[BASE URL: /proyectobase/api/services , API VERSION: 1.0.0]

6. Cómo generar un proyecto basado en Proyecto Base

El Proyecto Base está disponible a través de un arquetipo Maven (patrón que permite automatizar y parametrizar la creación de nuevos proyectos).

El proceso para la generación de un proyecto Java EE utilizando el arquetipo del proyecto base requiere tener instalado **OpenJDK 11** y **Maven 3.6 o superior**.

Los pasos a seguir son los siguientes:

1. Generar el fichero **[HOME]/.m2/settings.xml** (si no existe) y añadirle el siguiente contenido (bloque <profile> y <activeProfile>):

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
       https://maven.apache.org/xsd/settings-1.0.0.xsd">
       <profiles>
          <profile>
           <id>repositorigovernib</id>
           <repositories>
            <repository>
             <id>github-governib-maven</id>
             <name>GitHub GovernIB Maven Repository</name>
             <url>https://governib.github.io/maven/maven/</url>
            </repository>
           </repositories>
          </profile>
       </profiles>
       <activeProfiles>
          <activeProfile>repositorigovernib</activeProfile>
       </activeProfiles>
</settings>
```

2. Desde la línea de comandos, ejecutar el siguiente comando Maven (reemplazando el texto en negrita por el nombre del proyecto y el prefijo de la aplicación a generar):

Nota: Alternativamente, se puede acceder a la documentación del Proyecto Base en https://github.com/GovernIB/projectebase y utilizar el generador de proyectos para personalizar los datos resaltados.

```
set MAVEN_OPTS="-Dfile.encoding=UTF-8" && mvn org.apache.maven.plugins:maven-archetype-plugin:3.1.2:generate -B -DarchetypeGroupId=es.caib.projectebase -DarchetypeArtifactId=projectebase-archetype -DarchetypeVersion=1.0.1 -Dpackage=es.caib.projectebaseexemple -Dpackagepath=es/caib/projectebaseexemple -Dinversepackage=projectebaseexemple.caib.es -DgroupId=es.caib.projectebaseexemple -DartifactId=projectebaseexemple -Dversion=1.0.0 -Dprojectname=ProjecteBaseExemple -Dprojectnameuppercase=PROJECTEBASEEXEMPLE -Dprefix=pbe -Dprefixuppercase=PBE -DperfilBack=true -DperfilFront=true
```

Los parámetros -DperfilBack y -DperfilFront permiten especificar si se desea o no generar el módulo de backoffice y frontoffice, respectivamente.

Nota: En futuras versiones se perfeccionará y se añadirán nuevas funcionalidades como la integración con otras aplicaciones corporativas del GOIB.